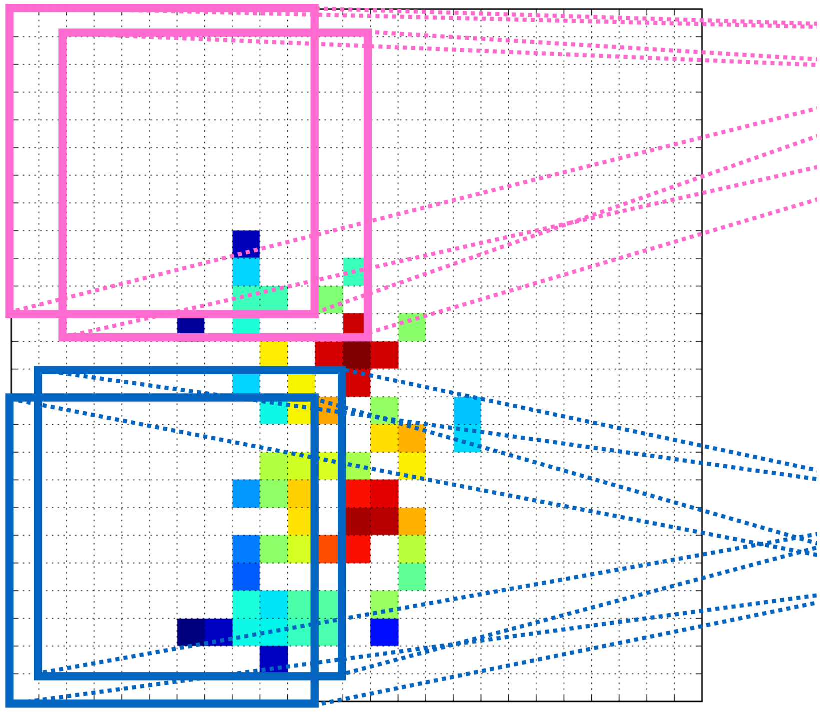
E Pluribus Unum Ex Machina
Learning from Many Collider Events at Once

*~ work with **Jesse Thaler** in PRD 103 (2021) 116013, 2101.07263 ~*

Benjamin Nachman

Lawrence Berkeley National Laboratory

bpnachman.com
bpnachman@lbl.gov
@bpnachman bnachman



BERKELEY LAB

BIDS
BERKELEY INSTITUTE FOR DATA SCIENCE

CMS Machine Learning Town Hall

July 27, 2021

Outline

- The core idea of this paper: *you can't (formally) gain by learning with multiple events at once, $f = f(x_1, \ldots, x_N)$.*

    ➡ This may be obvious to some of you and it may violate the intuition of others.

    ➡ There may be practical situations where it is better to train a classifier/regressor to process multiple events at once, especially if training can be made simpler.

- In addition to explaining the core idea, I'll describe some other statistical facts that may be useful for classification, inference, etc.

The statistical structure of collisions

To a very good approximation, collisions are independent:

$$p(\{x_1, \ldots, x_N\} \,|\, \theta) = \prod_{i=1}^{N} p(x_i \,|\, \theta)$$

x_i are features to represent events
(could be low-level or high-level)

The statistical structure of collisions

This means that the **per-ensemble likelihood ratio can be written as a product of per-event likelihood ratios.**

$$\frac{p(\{x_1,\ldots,x_N\}\,|\,\theta_A)}{p(\{x_1,\ldots,x_N\}\,|\,\theta_B)} = \prod_{i=1}^{N} \frac{p(x_i\,|\,\theta_A)}{p(x_i\,|\,\theta_B)}$$

This explains the informational equivalence of an optimal per-ensemble learning (left) and a per-event learning (right)

Collections of N events have more info than single events, but a per-event classifier applied N times has access to the same info.

Brief review: per-instance classification

We start with a loss function(al):

$$L[f] = -\sum_{i\in\text{class A}} A(f(x_i)) - \sum_{i\in\text{class B}} B(f(x_i))$$

It is often useful to consider the continuum limit:

$$L[f] = -\int dx\,\Big(p(x\,|\,\theta_A)A(f(x)) + p(x\,|\,\theta_B)B(f(x))\Big)$$

Typically, *A = log* and *B = 1-log*

Brief review: per-instance classification

Functional optimization shows that

$$-\frac{B'(f(x))}{A'(f(x))} = \frac{p(x\,|\,\theta_A)}{p(x\,|\,\theta_B)} \quad \longleftarrow \quad \text{optimal classifier}$$

This means that there are many loss functionals (defined by *A* and *B*) which result in an optimal classifier.

in particular, as long as *B'(f)/A'(f)* is a monotonic rescaling of *f* and the overall loss is convex

Brief review: per-instance classification

$$-\frac{B'(f(x))}{A'(f(x))} = \frac{p(x\,|\,\theta_A)}{p(x\,|\,\theta_B)}$$

← optimal classifier

Since it simplifies much of the notation, consider the “maximum likelihood classifier” (MLC) loss* with *A(f) = log(f)* and *B(f) = 1-f*. Then,

$$\mathrm{argmin}_f L_{\mathrm{MLC}}[f] = \frac{p(x\,|\,\theta_A)}{p(x\,|\,\theta_B)}$$

The loss value itself is the KL divergence between the conditional probabilities

*This was introduced in its exponential form by R. T. D’Agnolo and A. Wulzer in PRD 99 (2019) 015014, 1806.02350

Per-ensemble classification

One could insert *N* events instead of one into the MLC loss:

$$L_{\mathrm{MLC}}[f_N] = -\int d^N x \left(\overrightarrow{x} \,|\, \theta_A) \log f_N(\overrightarrow{x}) + p(\overrightarrow{x} \,|\, \theta_B)(1 - f_N(\overrightarrow{x})) \right)$$

Unsurprisingly,

$$\mathrm{argmin}_{f_N} L_{\mathrm{MLC}}[f_N] = \frac{p(\overrightarrow{x} \,|\, \theta_A)}{p(\overrightarrow{x} \,|\, \theta_B)}$$

Per-ensemble classification

$$\mathrm{argmin}_{f_N} L_{\mathrm{MLC}}[f_N] = \frac{p(\overrightarrow{x}\,|\,\theta_A)}{p(\overrightarrow{x}\,|\,\theta_B)}$$

However, we can also build this up from one event:

$$f_{1\to N} \equiv \prod_{i=1}^{N} f_1(x_i) \to \frac{p(\overrightarrow{x}\,|\,\theta_A)}{p(\overrightarrow{x}\,|\,\theta_B)}$$

If x is k-dimensional, then f_1 only requires samples of **dimension *k*** while f_N requires samples of **dimension k^N**

Per-ensemble classification

$$f_{1\to N} \equiv \prod_{i=1}^{N} f_1(x_i) \to \frac{p(\vec{x}\,|\,\theta_A)}{p(\vec{x}\,|\,\theta_B)}$$

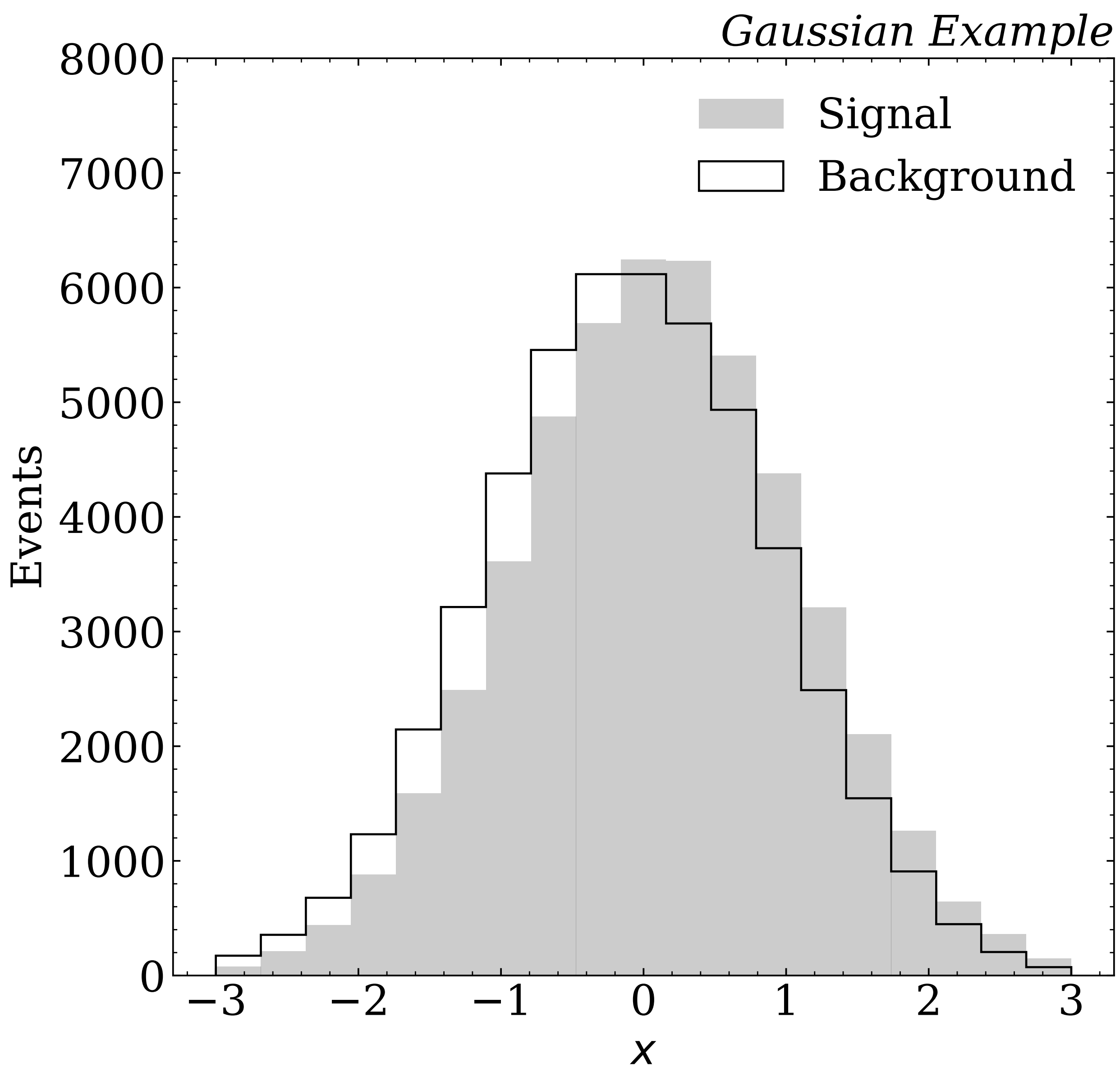
It is also possible to go in the converse direction:

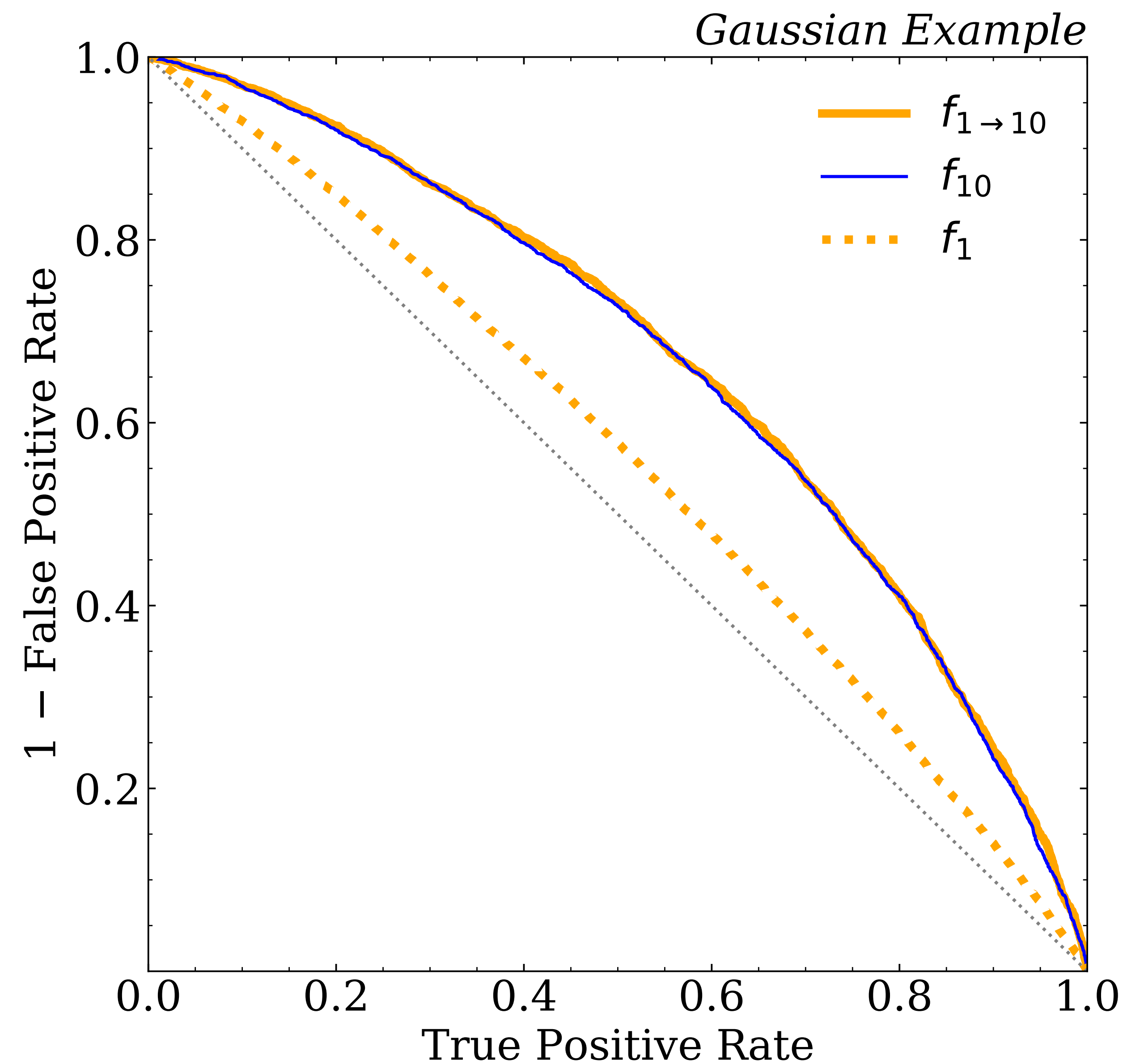
$$\tilde{f}_N(\vec{x}) \equiv \prod_{i=1}^{N} f_{N\to 1}(x_i)$$

Where the entire righthand side is used to minimize the ensemble MLC loss. This construction is at the core of set-based neural networks like **Deep Sets**.

M. Zaheer et al. NeurIPS 2017

Examples: Classification [Gaussian]



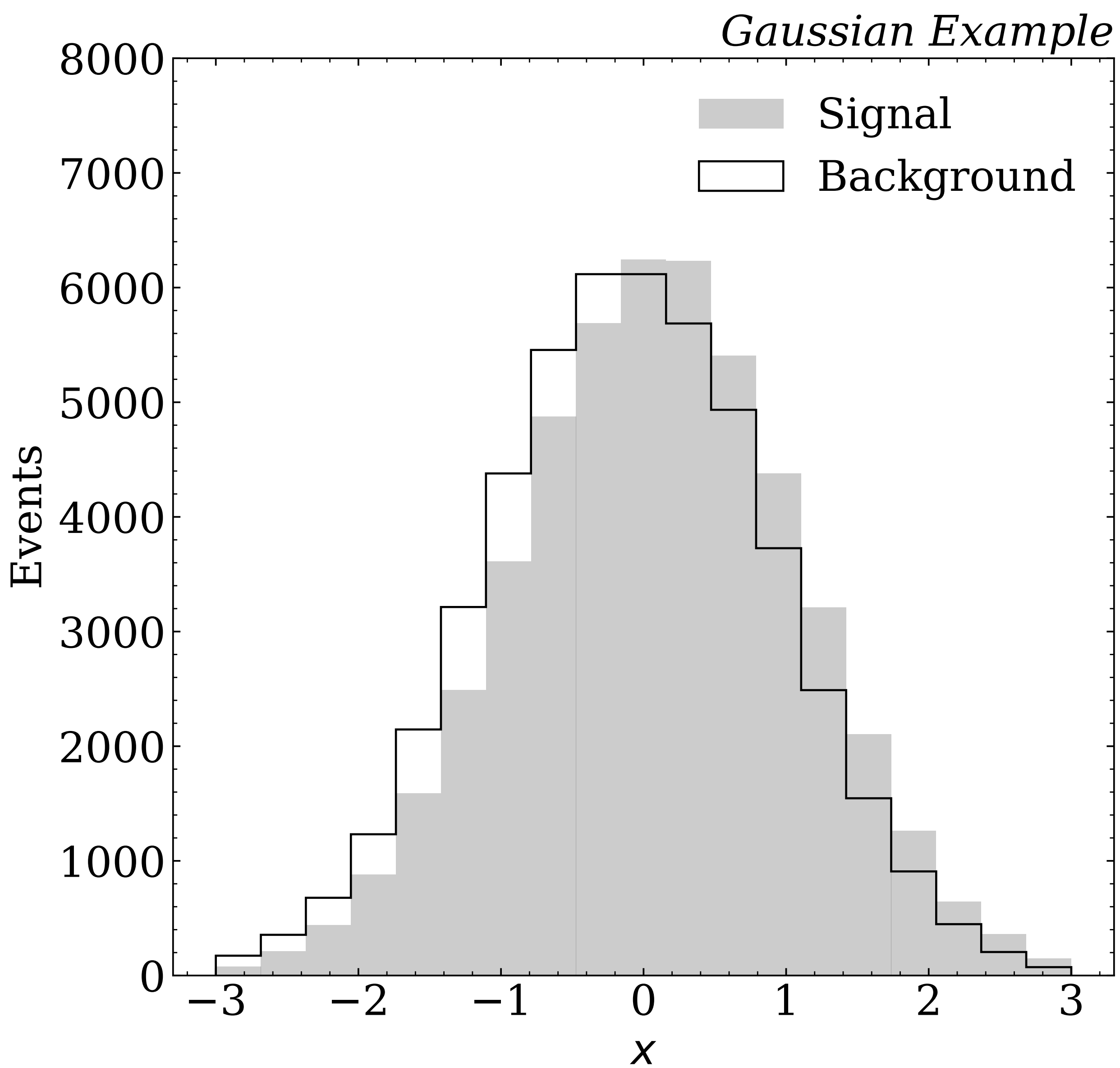


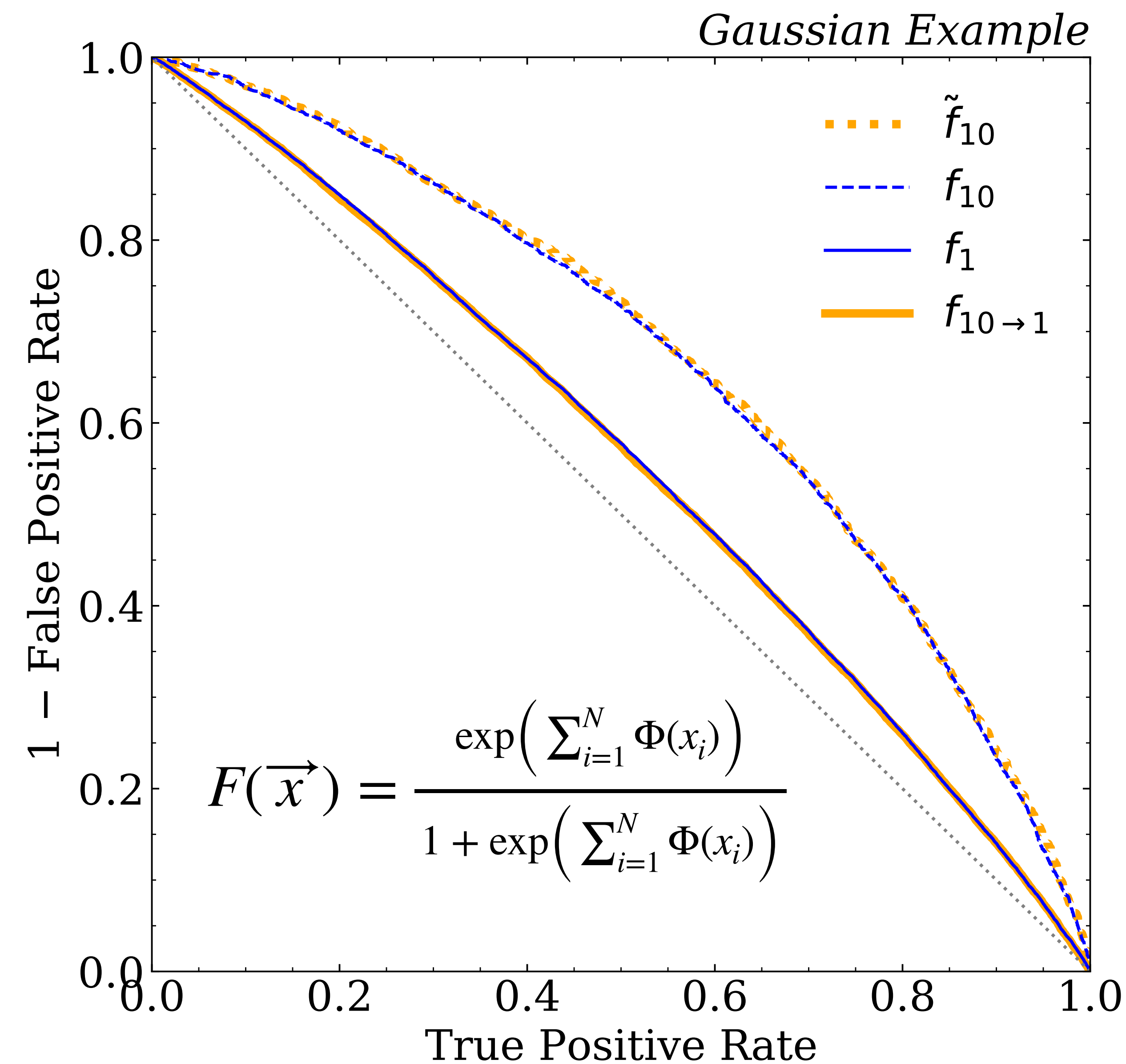
$f_{1\rightarrow 10}$ per-event classifier applied 10 times

f_{10} per-(10-event) classifier

f_1 per-event classifier

Examples: Classification [Gaussian]

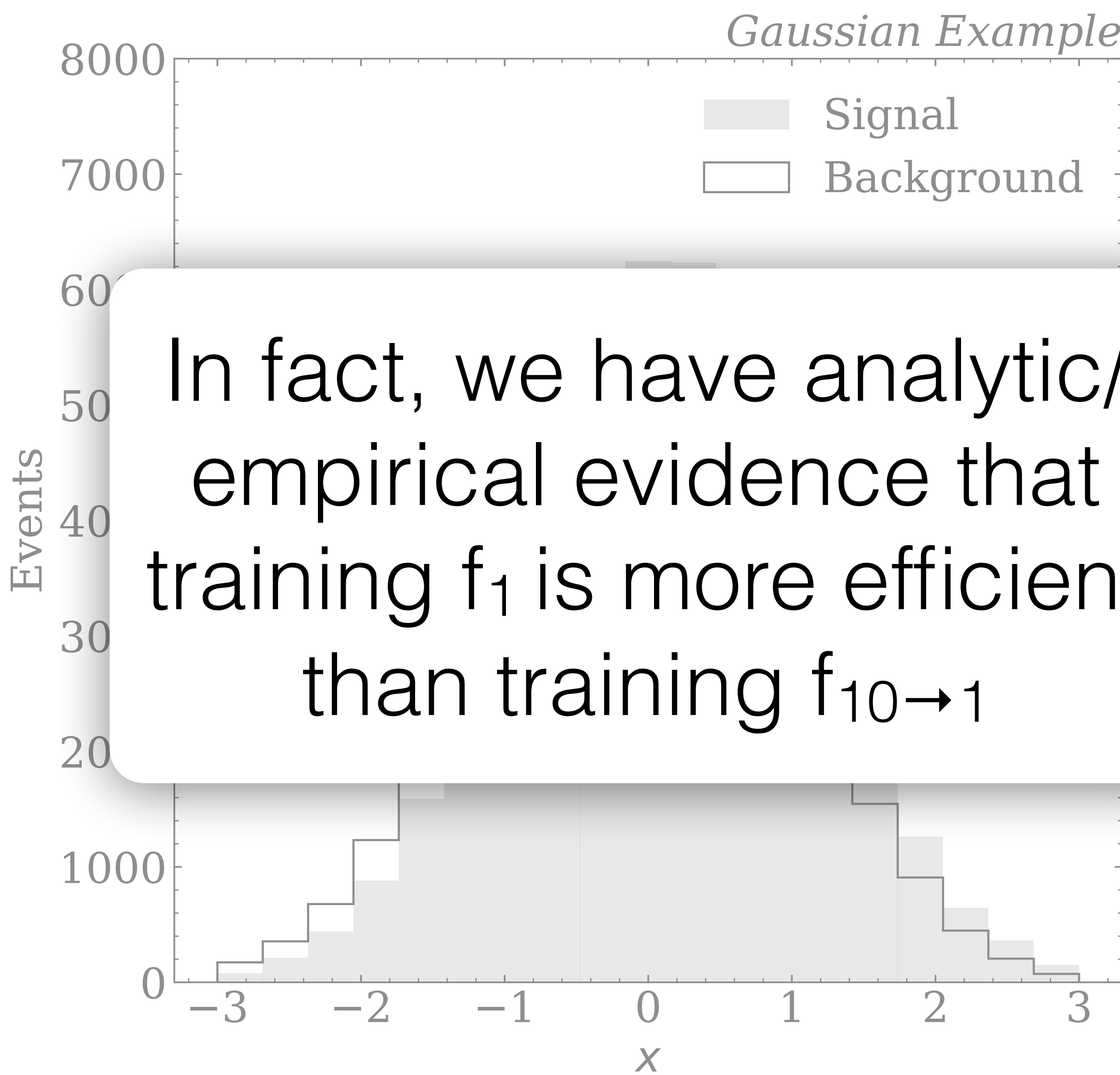




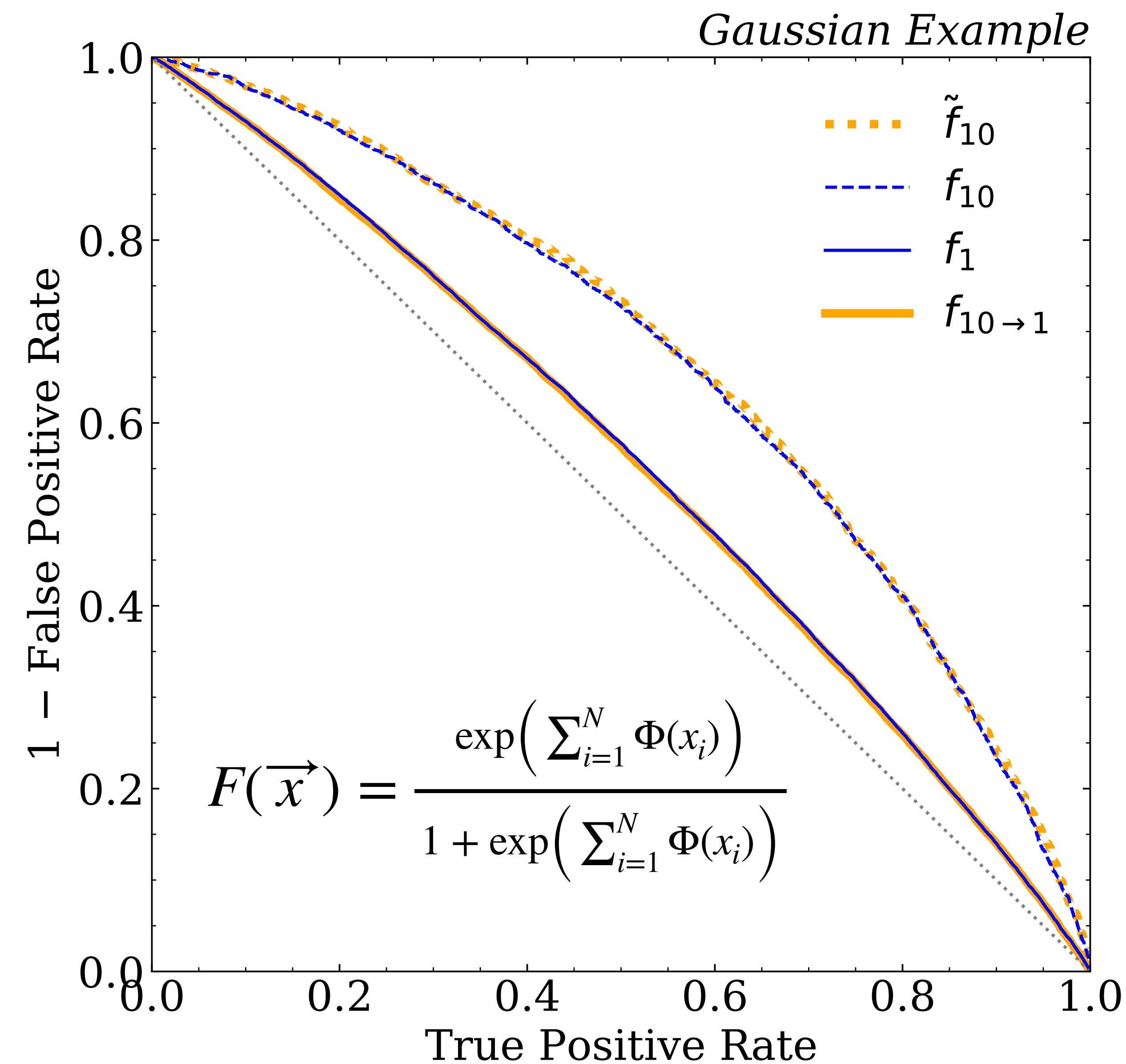
$f_{10\to1}$ Train a **particle flow network** *F(Φ(x))* with latent space dimension 1 and non-trainable *F*

P. Komiske, E. Metodiev, J. Thaler, JHEP 01 (2019) 121, 1810.05165

Examples: Classification [Gaussian]



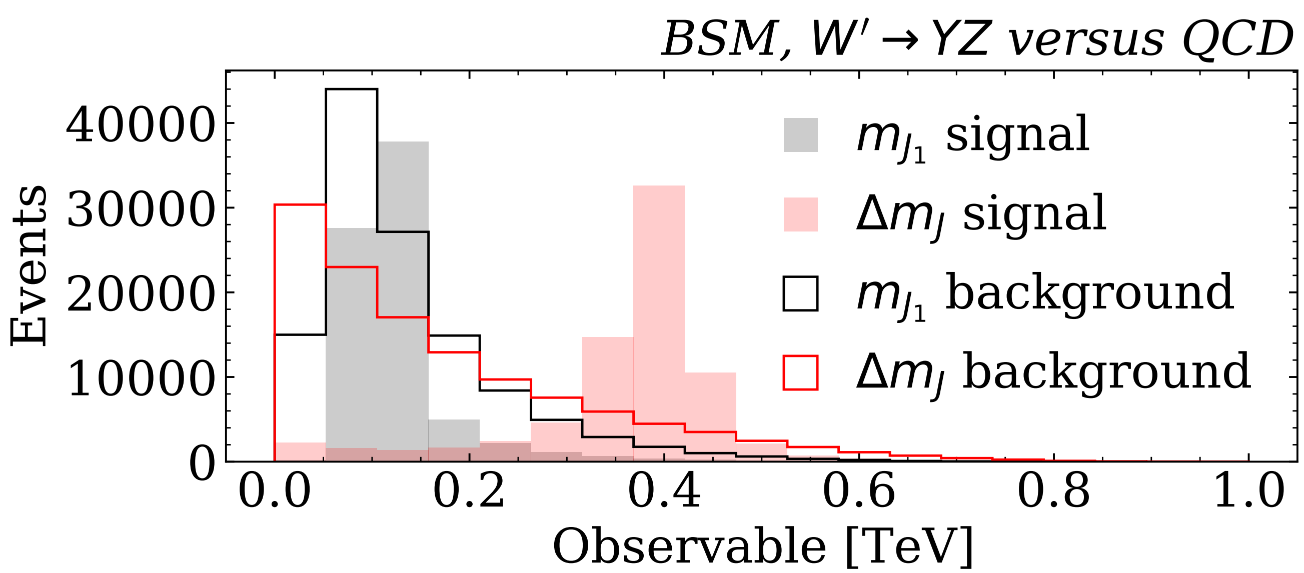
In fact, we have analytic/ empirical evidence that training f_1 is more efficient than training $f_{10\to1}$

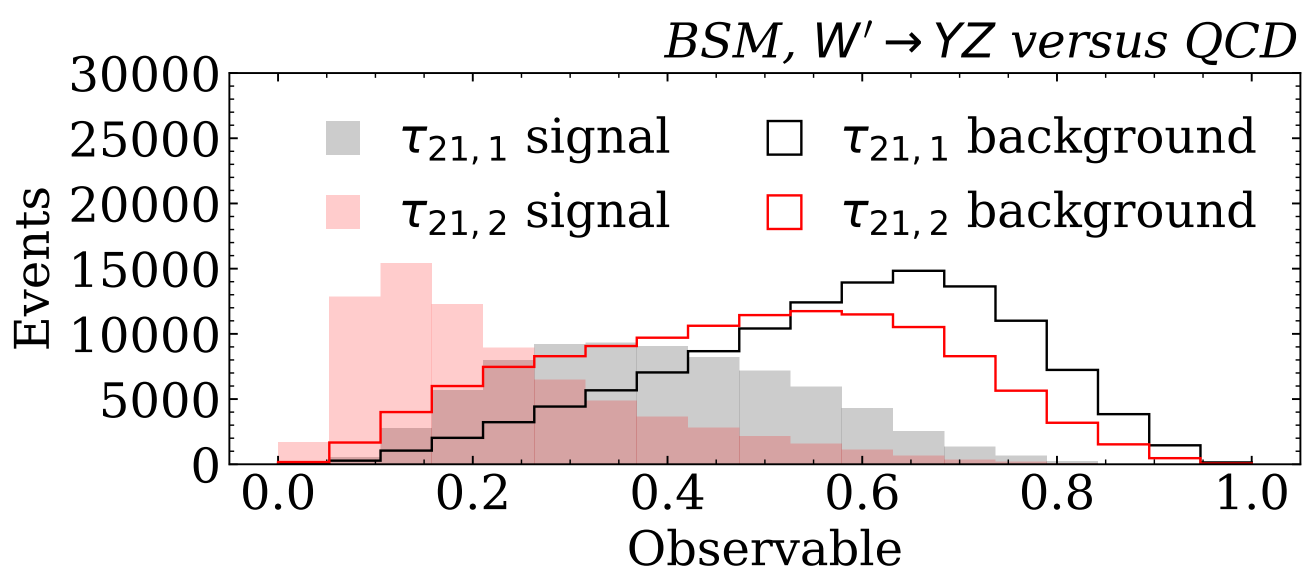


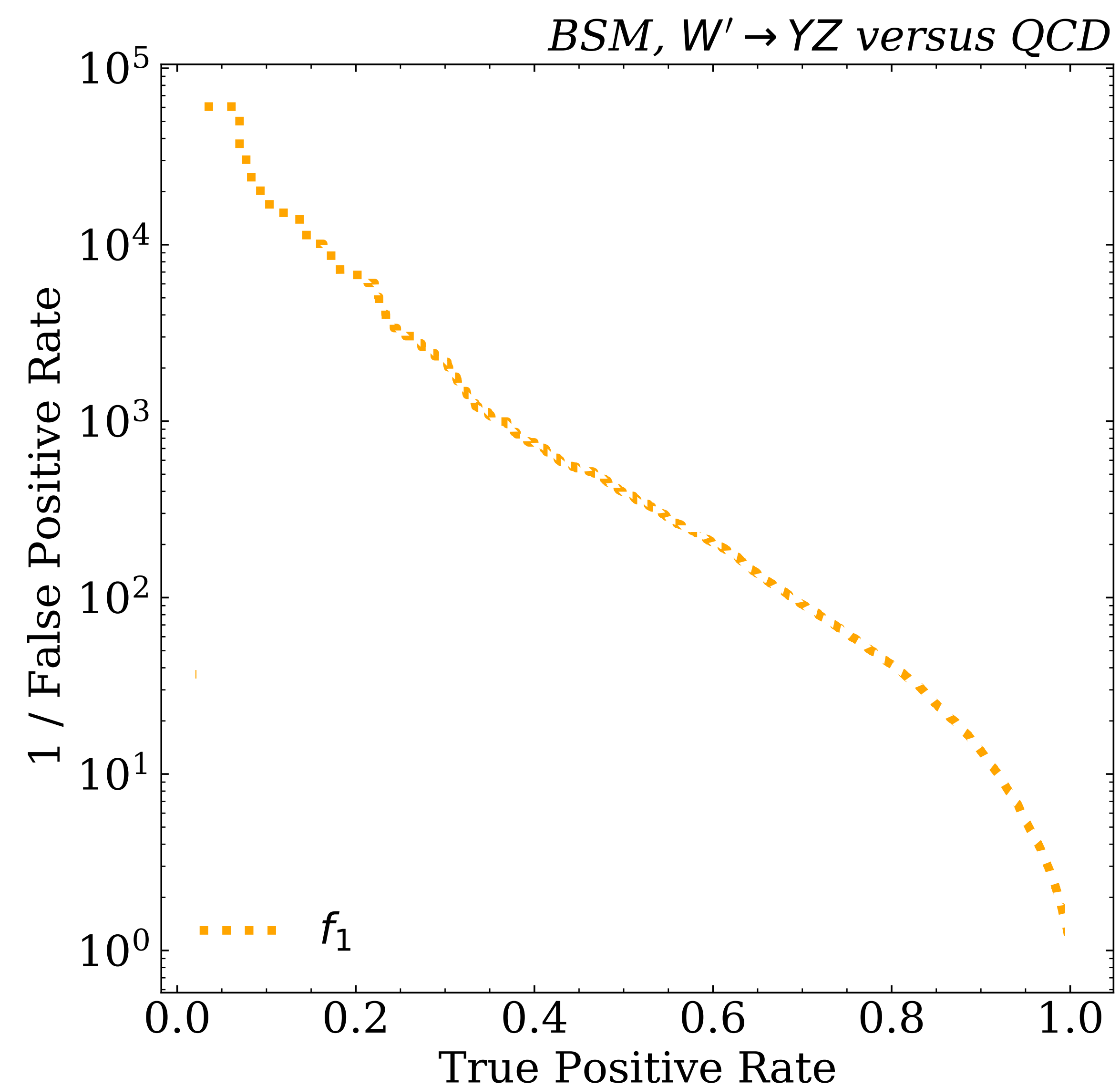
$f_{10\to1}$ Train a **particle flow network** *F(Φ(x))* with latent space dimension 1 and non-trainable *F*

P. Komiske, E. Metodiev, J. Thaler, JHEP 01 (2019) 121, 1810.05165

Examples: Classification [BSM]

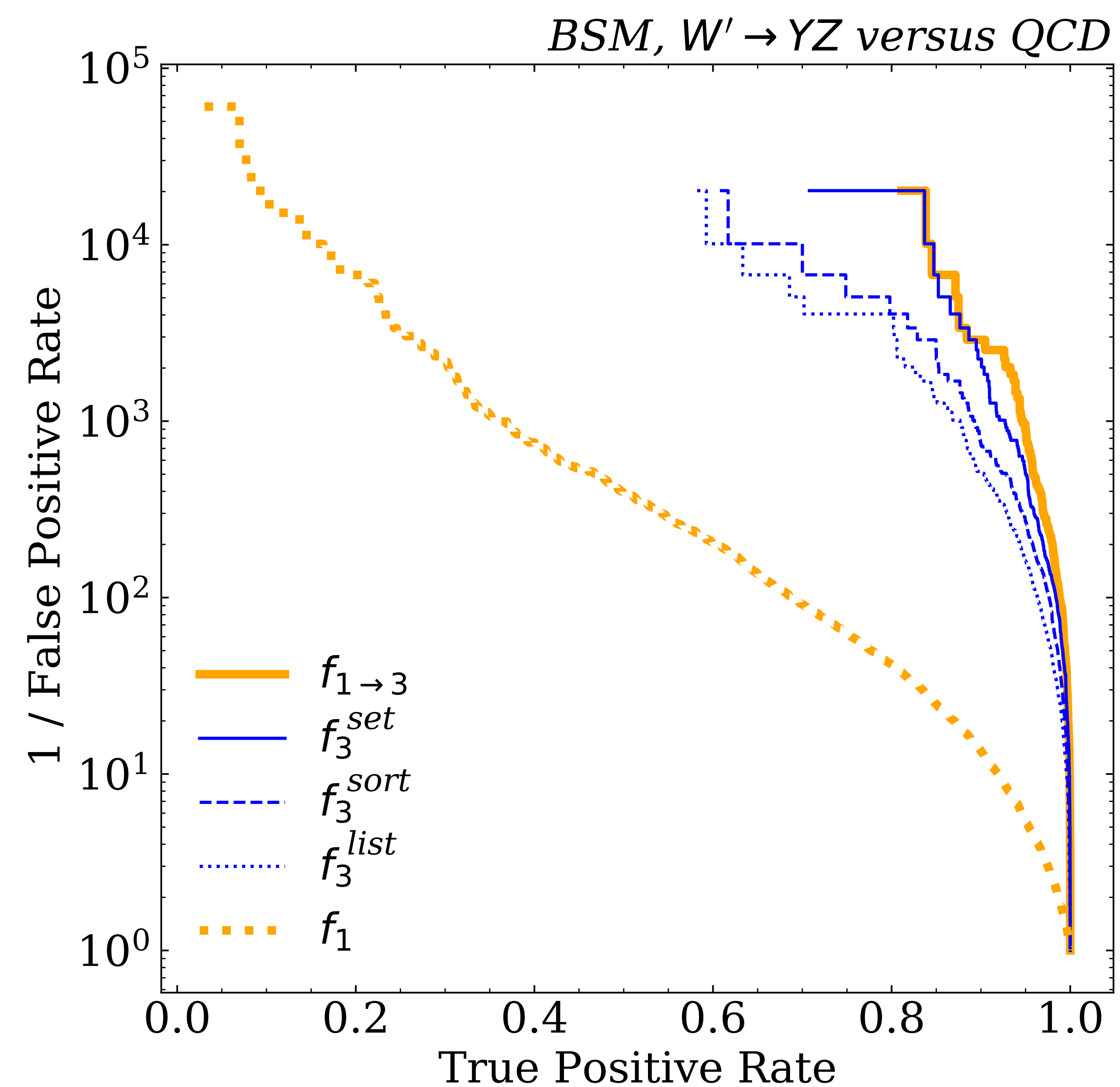






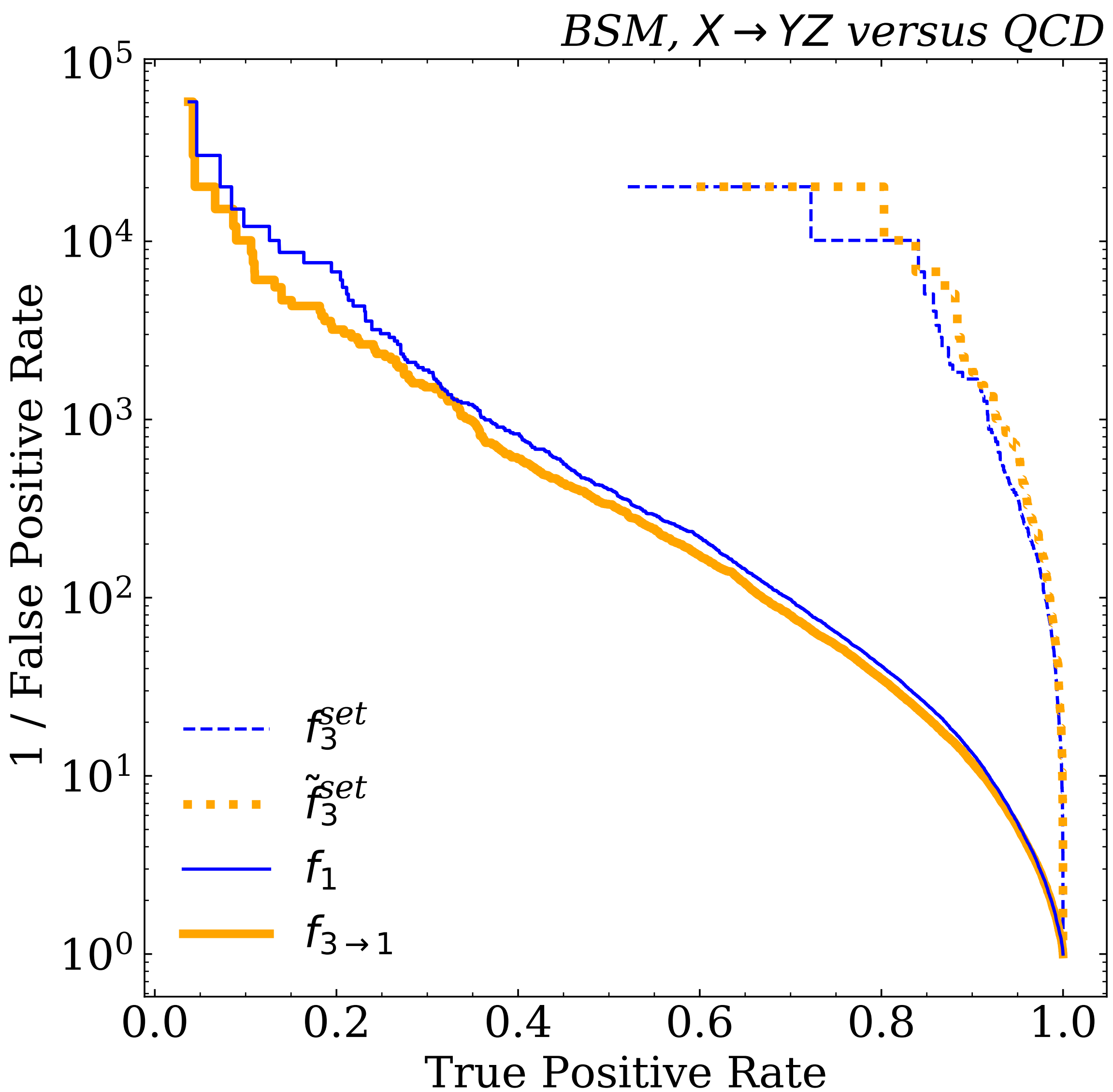
Examples: Classification [BSM]

(solid orange line)	$f_{1\to 3}$	per-event classifier applied 3 times
(solid blue line)	f_3^{set}	per-3-event classifier, using PFN across events
(dashed blue line)	f_3^{sort}	per-3-event classifier, events sorted by m_{J_1}
(dotted blue line)	f_3^{list}	per-3-event classifier
(dotted orange line)	f_1	per-event classifier



Examples: Classification [BSM]

(orange solid line)	$f_{1\rightarrow 3}$	per-event classifier applied 3 times
(blue solid line)	f_3^{set}	per-3-event classifier, using PFN across events
(blue dashed line)	f_3^{sort}	per-3-event classifier, events sorted by m_{J_1}
(blue dotted line)	f_3^{list}	per-3-event classifier
(orange dotted line)	f_1	per-event classifier



$\tilde{f}_3^{set}$ 1D latent space, Non-learnable F

Per-ensemble regression I

Most common strategy is maximum likelihood:

$$\theta_{\mathrm{ML}} = \mathrm{argmax}_{\theta}\, p(\overrightarrow{x} \,|\, \theta)$$

(see paper for other examples beyond regression, e.g. estimating the mutual information)

Per-ensemble regression I

Most common strategy is maximum likelihood:

$$\theta_{\mathrm{ML}} = \mathrm{argmax}_\theta p(\overrightarrow{x} \,|\, \theta)$$

One way of doing this would be to train a **parameterized classifier** using the MLC loss

$$f(x,\theta) = \frac{p(x\,|\,\theta)}{p(x\,|\,\theta_0)} \rightarrow \theta_{\mathrm{ML}} = \mathrm{argmin}_\theta \left\{ -\sum_{i=1}^{N} \log f(x_i,\theta) \right\}$$

fixed reference

K. Cranmer, J. Pavez, G. Louppe, 1506.02169; P. Baldi et al. EPJC 76 (2016) 235, 1601.07913

Per-ensemble regression II

Step 1 Step 2

$$f(x,\theta) = \frac{p(x\,|\,\theta)}{p(x\,|\,\theta_0)} \rightarrow \theta_{\mathrm{ML}} = \mathrm{argmin}_\theta \left\{ -\sum_{i=1}^{N} \log f(x_i,\theta) \right\}$$

This requires two steps (amortized). Can we do it in one step?

$$\mathrm{argmax}_\theta \left\{ \min_f L_{\mathrm{MLC}}[f] \right\} = \theta_{\mathrm{ML}}$$

(see paper for derivation)

This does it all in one step, but does require a minimax optimization

Per-ensemble regression III

You may also ask, why not directly regress θ from N events?

$$L_{\text{MSE}}[g_N] = -\int d^N x \, p(\overrightarrow{x}, \theta)\left(g_N(\overrightarrow{x}) - \theta\right)^2$$

MSE = mean squared error

This is prior-dependent, but it is well-known that formally:

$$g_N(\overrightarrow{x}) = \langle \theta \,|\, \overrightarrow{x} \rangle$$

Per-ensemble regression III

$$g_N(\overrightarrow{x}) = \langle \theta | \overrightarrow{x} \rangle$$

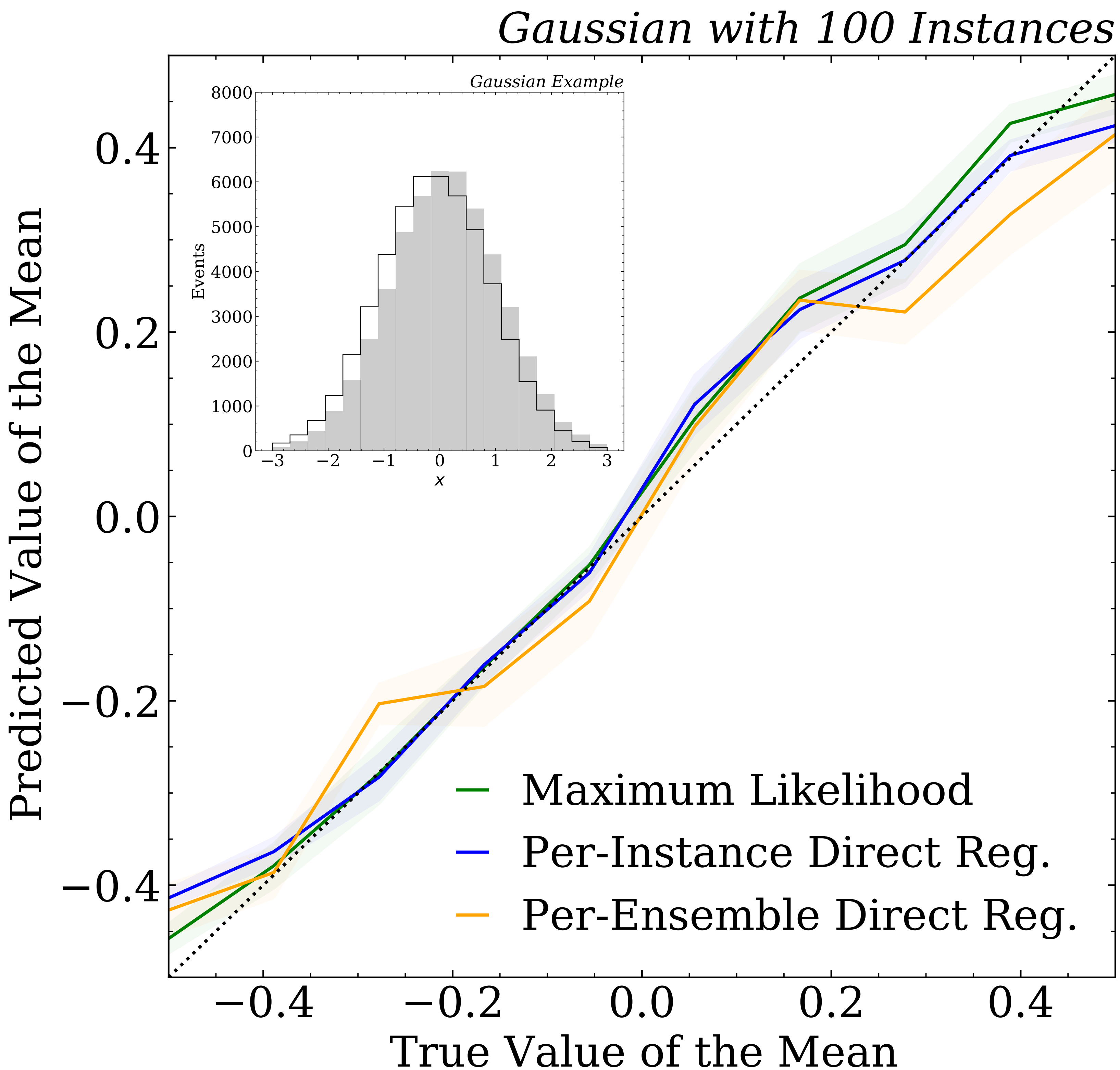
$$g_N(\overrightarrow{x}) = \int d\theta\, \theta\, p(\theta | \overrightarrow{x})$$

$$= \int d\theta\, \theta\, \frac{p(\overrightarrow{x} | \theta)\, p(\theta)}{p(\overrightarrow{x})}$$

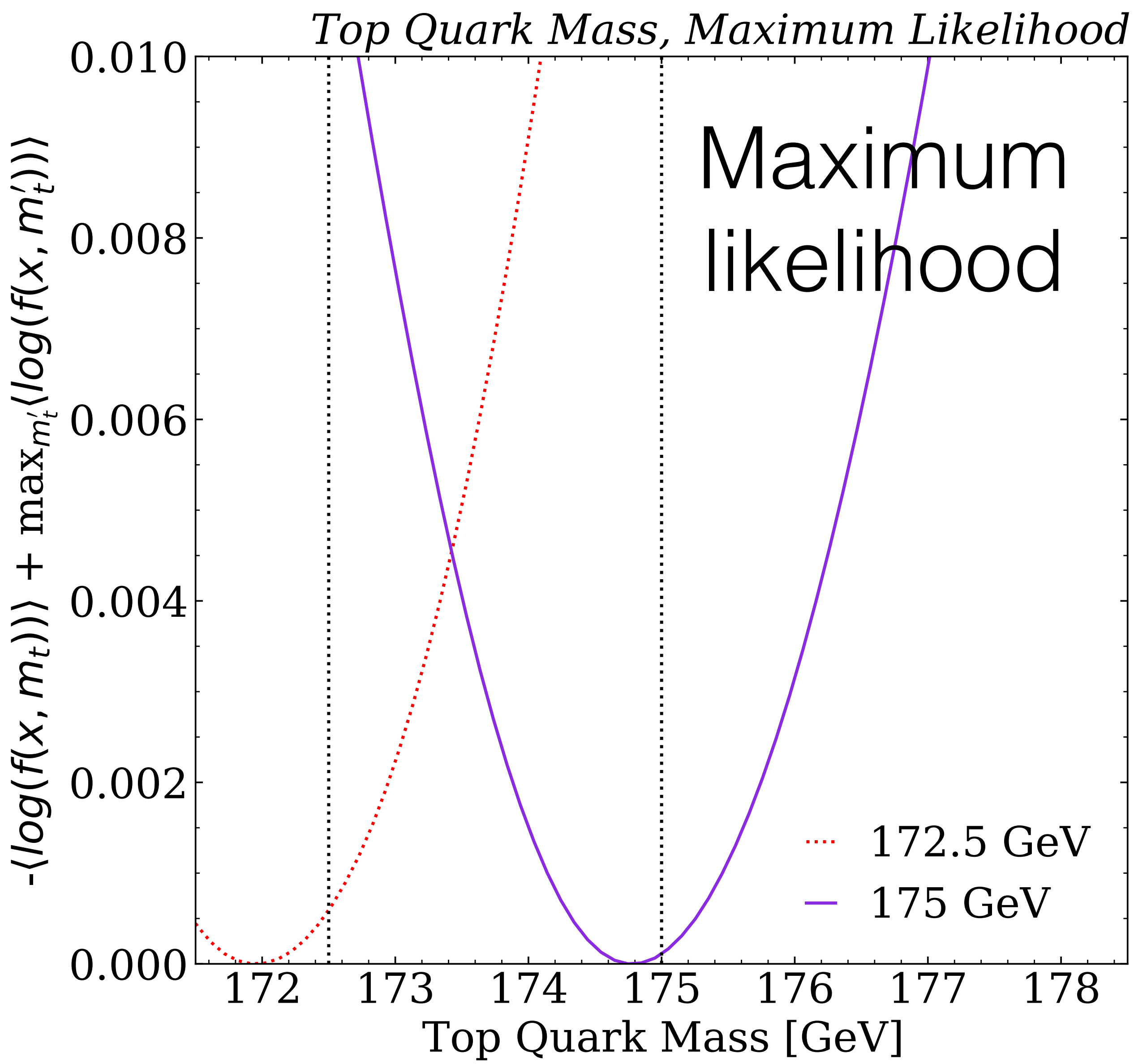
$$= \int d\theta\, \theta\, \frac{\frac{p(\overrightarrow{x} | \theta)}{p(\overrightarrow{x} | \theta_0)}\, p(\theta)}{\int d\theta' \frac{p(\overrightarrow{x} | \theta')}{p(\overrightarrow{x} | \theta_0)}\, p(\theta')} = \int d\theta\, \theta\, \frac{f_N(\overrightarrow{x}, \theta)\, p(\theta)}{\int d\theta' f_N(\overrightarrow{x}, \theta')\, p(\theta')}$$

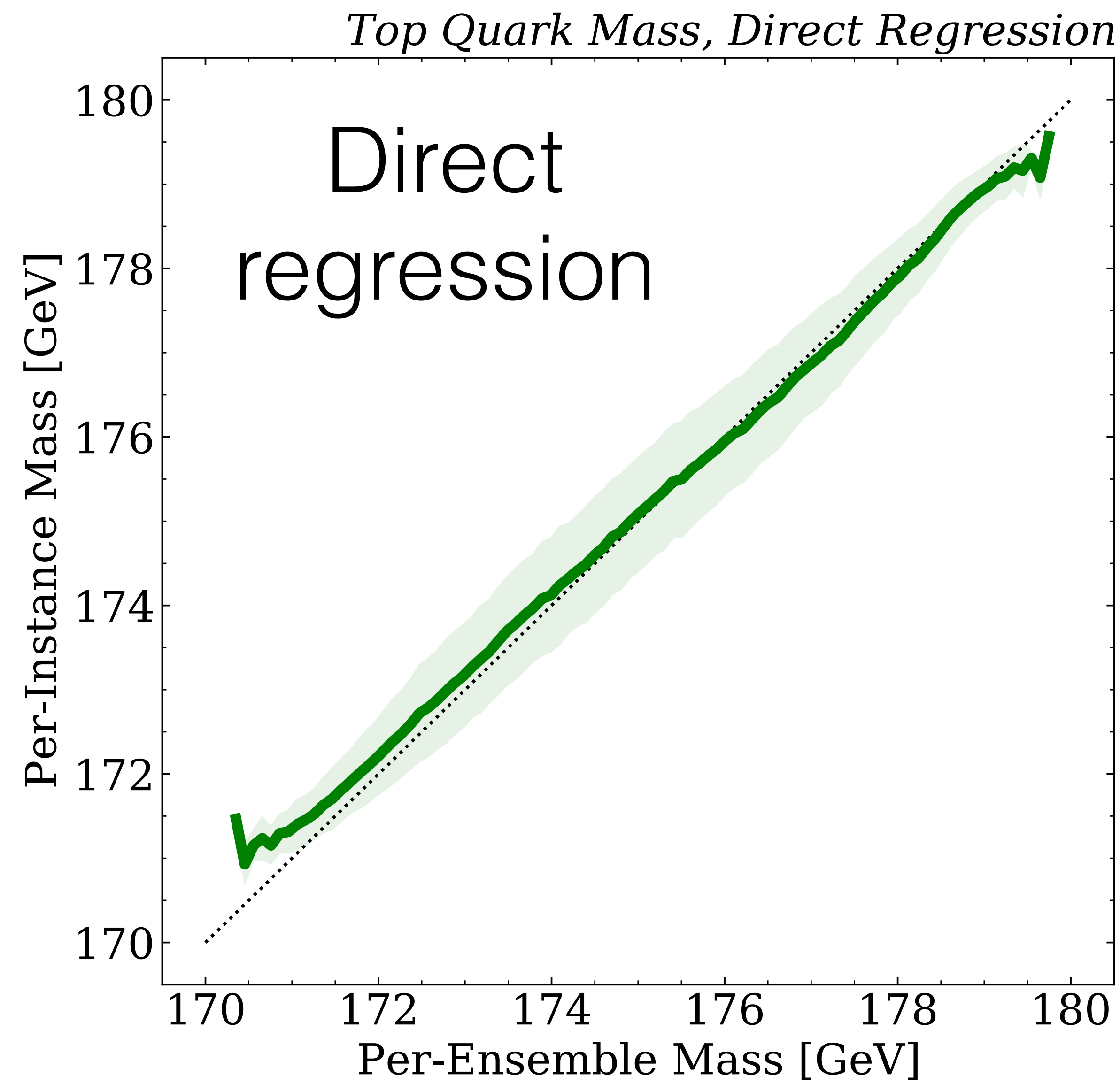
i.e. this is still secretly per-event classification !

Examples: Regression [Gaussian]



Examples: Regression [top quarks]

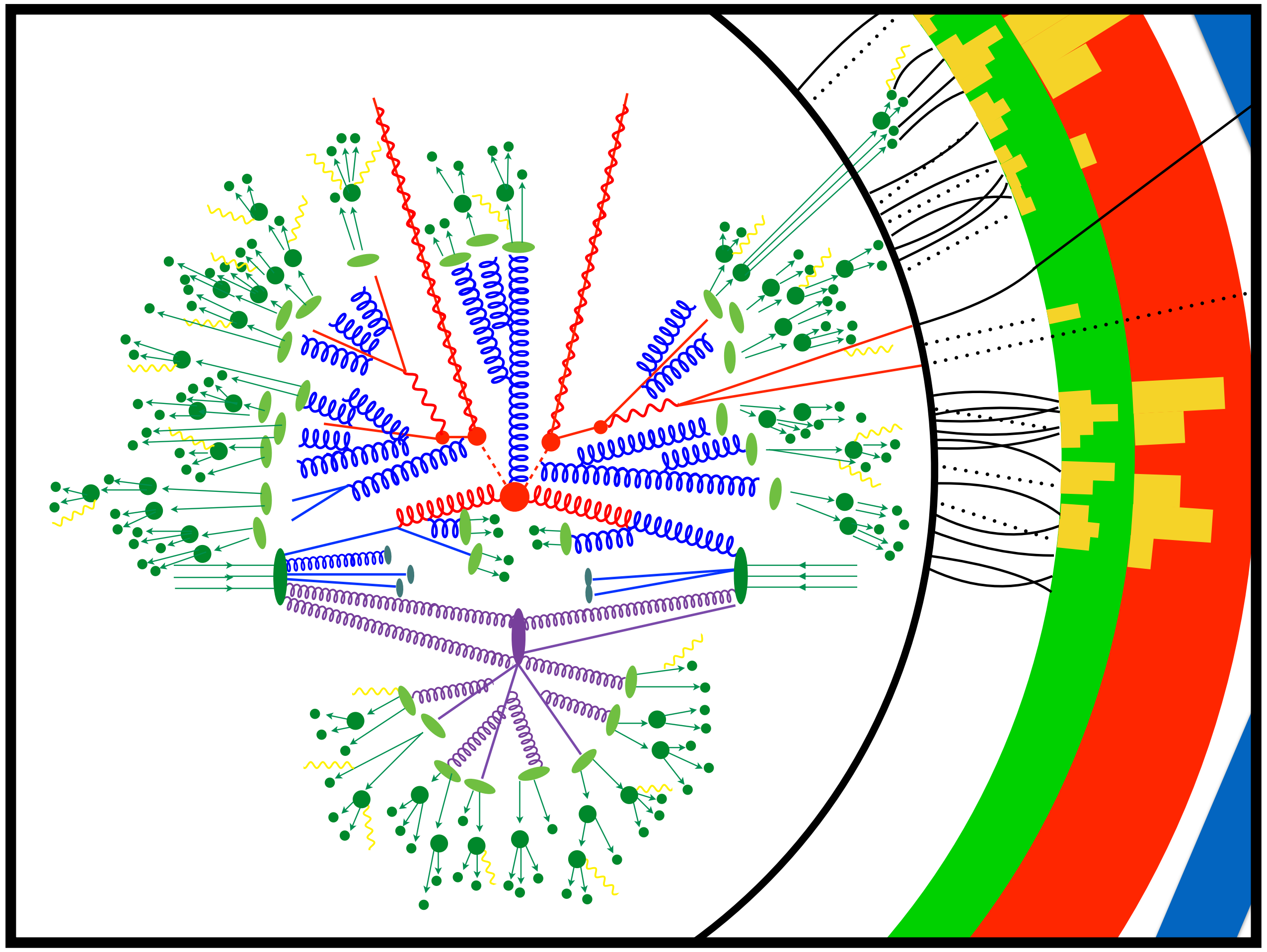




In both cases, the per-instance approaches give the same result as the per-ensemble approach

Conclusions and outlook

Today, I've told you about the interplay between per-event and per-ensemble learning



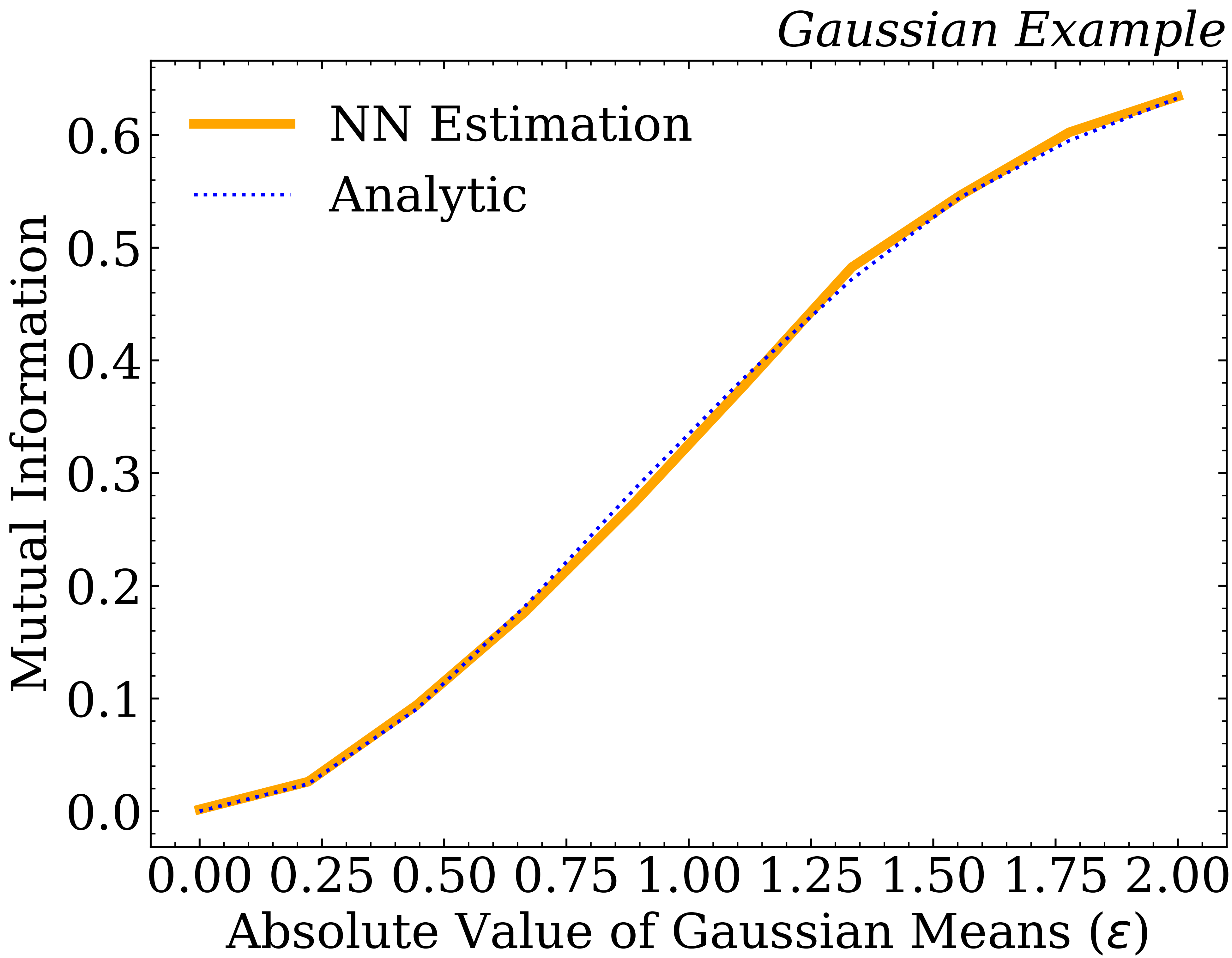
Formally, these are equivalent. Per-event models are less complex, but there may be practical reasons to prefer one over the other

GitHub.com/bnachman/EnsembleLearning

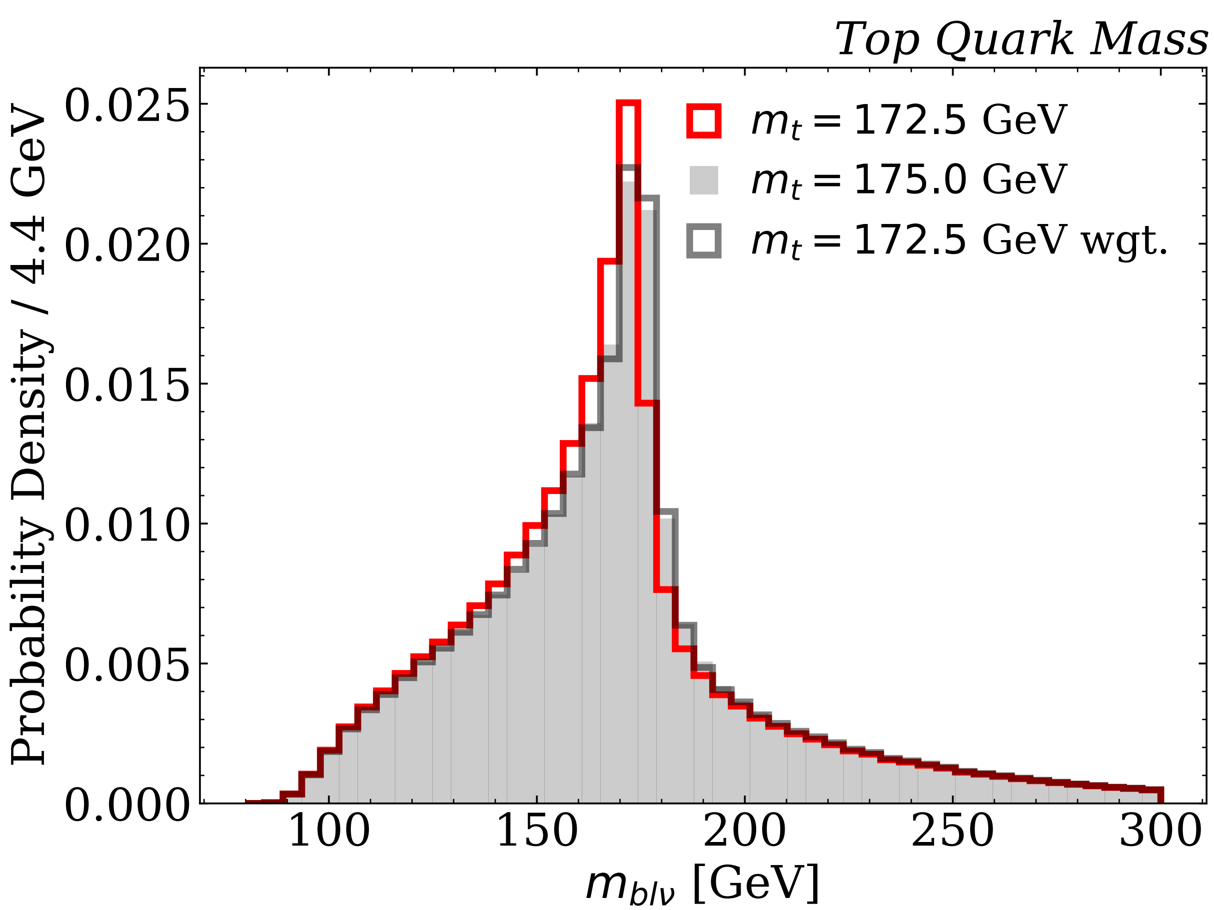
B. Nachman and J. Thaler in PRD 103 (2021) 116013, 2101.07263

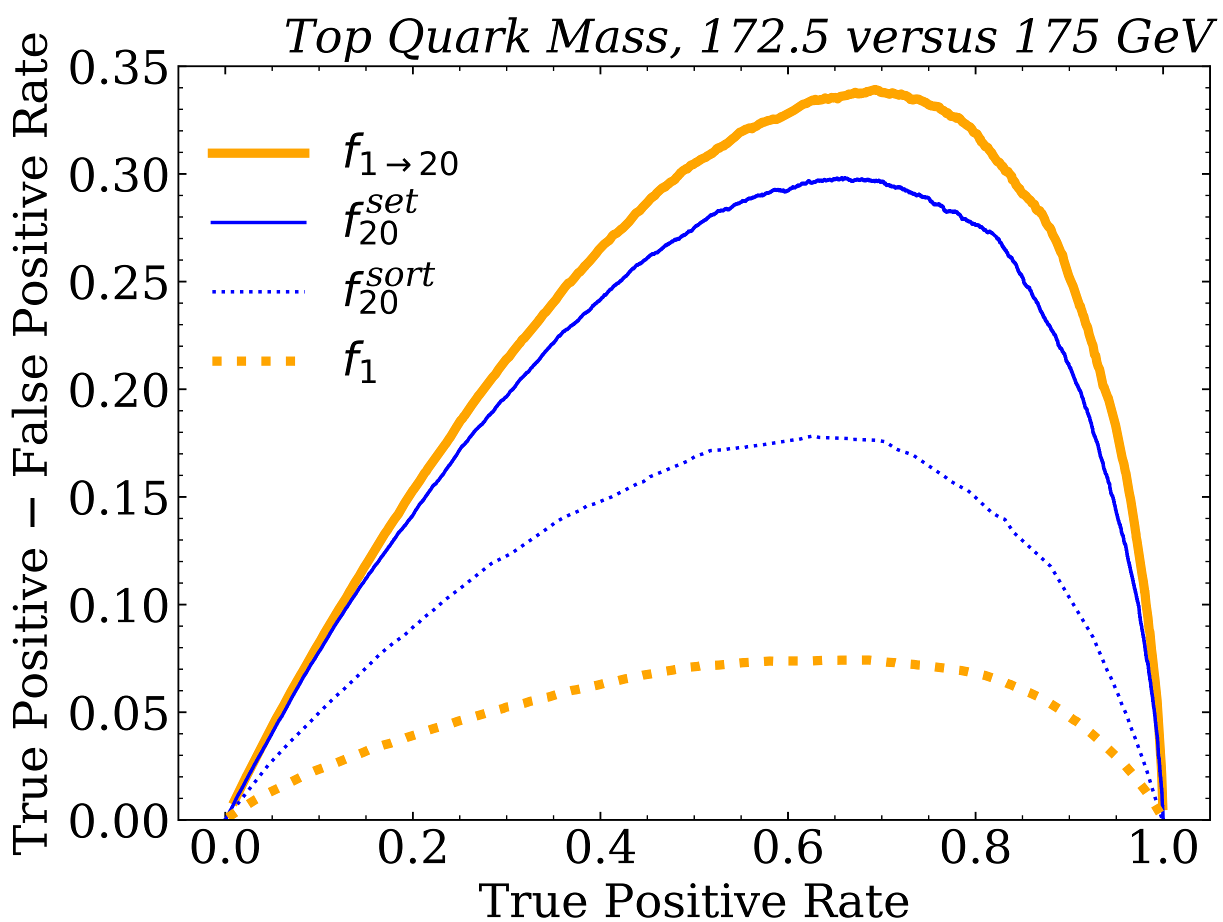
Backup

Estimating the Mutual Information



Examples: Classification [top quarks]





Examples: Classification [top quarks]